

stepwiseCM: Stepwise Classification of Cancer Samples using High-dimensional Data Sets

Askar Obulkasim

Department of Epidemiology and Biostatistics, VU University Medical Center
P.O. Box 7075, 1007 MB Amsterdam, The Netherlands
askar703@gmail.com

Contents

1 Introduction

This vignette shows the use of the **stepwiseCM** package for the integrative analysis of two heterogeneous data sets. For more detailed information on the algorithm and assumptions we refer to the article (?) and its supplementary material. Throughout this vignette we use low dimensional clinical covariates and high-dimensional molecular data (gene expression) to illustrate the use and feature of the package **stepwiseCM**. Also, we use clinical covariates at the first stage and expression data at the second stage. But the package can also be used in a setting where both data sets are high-dimensional data. User also has freedom to decide the order of the data sets use in different stages.

The following features are discussed in detail:

- Prediction with user defined algorithm.
- Proximity matrix calculation.
- Reclassification score (RS) calculation.
- Reference curve generation.
- Assigning test samples to one of the two data types.

These are illustrated on an example data set, which is introduced first.

2 Central Nervous System cancer data (CNS)

The CNS tumor data (?) has been used to predict the response of childhood malignant embryonal tumors of CNS to the therapy. The data set is composed of 60 patients (biopsies were obtained before they received any treatment), 21 patients died and 39 survived within 24 months. Gene expression data has 7128 genes and clinical features are Chang stage (nominal), sex (binary), age (nominal), chemo Cx (binary), chemo VP (binary).

Data can be downloaded as an .RData file from <http://mixology.qfab.org/lecao/package.html>.

```
> # load the full Central Nervous System cancer data
> library(stepwiseCM)
> data(CNS)
```

The code above loads a list object `CNS` with following components

- `mrna`: A matrix containing the expression values of 7128 genes (rows) in 60 samples (columns).
- `cli`: A matrix containing the five clinical risk factors: ChangStage, Sex, Age, chemo Cx, chemo VP.
- `class`: A vector of the class labels. Died patients are denoted with 0 and survived patients are denoted with 1.
- `gene.name`: A data frame containing the gene names and descriptions.

3 Prediction

The first step of an integrative analysis comprises obtaining the prediction labels of set training set using the two data types independently. The objective of this prediction step is to assess the classification power of the two data sets. The prediction labels of the given set will be attained by using the user selected classification algorithm. One would specify the type of algorithms via the `type` parameter. Currently, nine algorithms are available for user to choose from. Note that “TSP”, “PAM”, “`plsrf_x`” and “`plsrf_x_pv`” are exclusively designed for high-dimensional data. We can attain the prediction labels of the training set either by the Leave-One-Out-Cross-Validation (LOOCV) or the K-fold cross validations via the argument `CVtype`. If set to `k-fold`, the argument `outerkfold` (the number of folds) also has to be specified. If set to the `loocv`, algorithm ignore the argument `outerkfold`. The parameter `innerkfold` defines the number of cross validation used to estimate the model parameter(s) e.g. λ in “GLM_L1” (Goeman, 2011).

Note that, the type of algorithms can be used are not limited by the ones mentioned here. The output of this function will be used as input in later stage. Thus, if user already has predicted label of the set under consideration, this step can be skipped.

Here, we randomly split the data into two sets with equal sizes and use them as training and test set, respectively. we choose to use the “RF” option for clinical data and “`plsrf_x`” (?) option for expression data, respectively. For computational convenience we set `CVtype` = “k-fold”, `outerkfold` = 2 and `innerkfold` = 2. The optimal argument `test` is also given.

```
> # gain the prediction labels of the training set
> data(CNS)
> attach(CNS)
> #Note that columns of the train set should corresponding the sample and rows
> #corresponding the feature.
> id <- sample(1:ncol(mrna), 30, replace=FALSE)
> train.exp <- mrna[, id]
> train.cli <- t(cli[id, ])
> test.exp <- mrna[, -id]
> test.cli <- t(cli[-id, ])
> train.label <- class[id]
> test.label <- class[-id]
> pred.cli <- Classifier(train=train.cli, test=test.cli, train.label=train.label,
```

```

+                               type="RF", CVtype="k-fold", outerkfold=2, innerkfold=2)
> pred.exp <- Classifier(train=train.exp, test=test.exp, train.label=train.label,
+                               type="plsrf_x", CVtype="k-fold", outerkfold=2, innerkfold=2)
> # Classification accuracy of the training set from clinical data is:
> sum(pred.cli$P.train == train.label)/length(train.label)
> # Classification Accuracy of the training set from expression data is:
> sum(pred.exp$P.train == train.label)/length(train.label)

```

We can also obtain the prediction labels using the *Classifier.par* function if the multiple-core computation is desired. If this function is called, one can take advantage of its powerful parallel computation capability by specifying the *ncpus* option which controls the number of cores assigned to the computation.

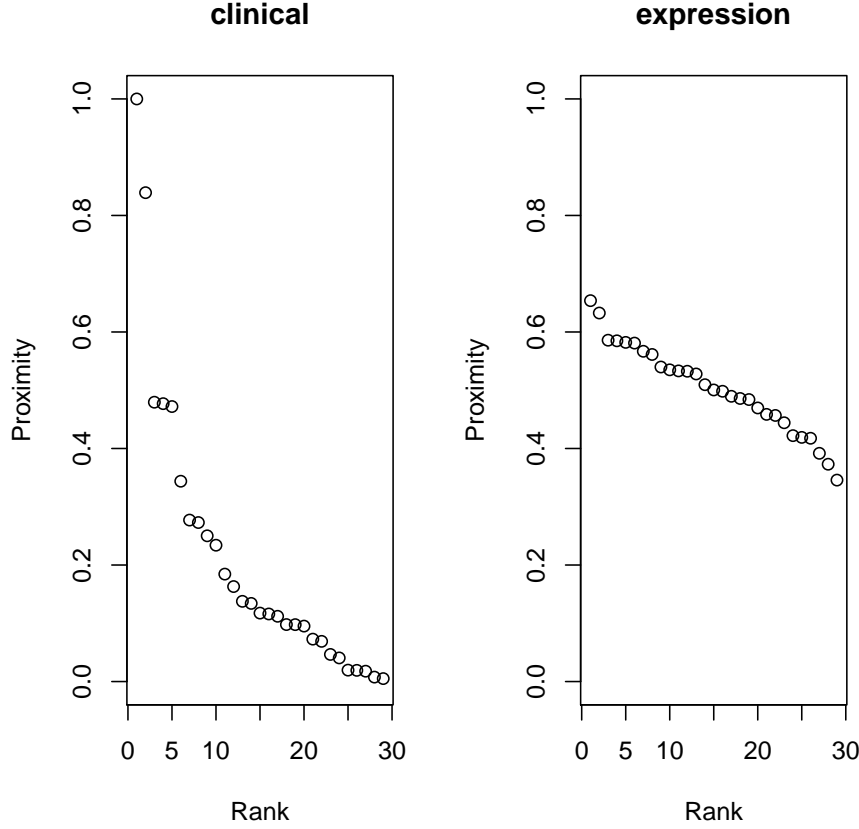
4 The proximity matrix calculation

The second step is to calculate the proximity matrix of the training set with the Random Forest algorithm (?). This matrix is used to locate the test sample's position in the data space used at the first stage (presumed that test samples have measurements only for this data type) and locate the "pseudo" position of the test sample in the data space used at the second stage by the indirect mapping (IM). The same as in the prediction step, we will calculate the two proximity matrices using the two data independently. Because of the large number of irrelevant features in high-dimensional data, the value of *N* need to be set large, so that final averaged proximity matrix will cancel out the irrelevant features effect. Here, we set *N* = 5 and calculation is done without the parallel processing procedure.

```

> prox.cli <- Proximity(train=train.cli, train.label=train.label, test=test.cli, N=5)
> prox.exp <- Proximity(train=train.exp, train.label=train.label, N=5)
> #check the range of proximity values from two different data types
> par(mfrow=c(1,2))
> plot(sort(prox.cli$prox.train[1, ][-1], decreasing=TRUE), main="clinical", xlab="Rank",
+       ylab="Proximity", ylim=c(0,1))
> plot(sort(prox.exp$prox.train[1, ][-1], decreasing=TRUE), main="expression", xlab="Rank",
+       ylab="Proximity", ylim=c(0,1))

```



The range of proximity values from two different data types can be visualised in following way: choose the first sample in the training set and order (descending) the rest of the samples proximity values (from the first data set) w.r.t this sample. Apply similar procedure to the proximity matrix from the second data set. As Figure shows, the two types of data we used produce the proximity values with different ranges. Large range is observed in the clinical data setting, small range is observed in the expression data setting. Note, since the tree construction inside the RF has randomness, the resulting proximity matrix is may not be the same in each run.

5 The reclassification score (RS) calculation

After finishing the first two steps, we received all the inputs required for the calculation of the RS. The RS can be attained by using the *RS.generator* function. For this function, we only need to specify which type of similarity measure is should be used in during the calculation. If type = “proximity”, then RS is calculated directly from the proximity values. If type = “rank”, function first order the proximity values and uses their ranking instead. Latter one is more robust against the different ranges problem as we observed previously. Here, in order to see the differences in the outputs from two methods, we set type = “both”.

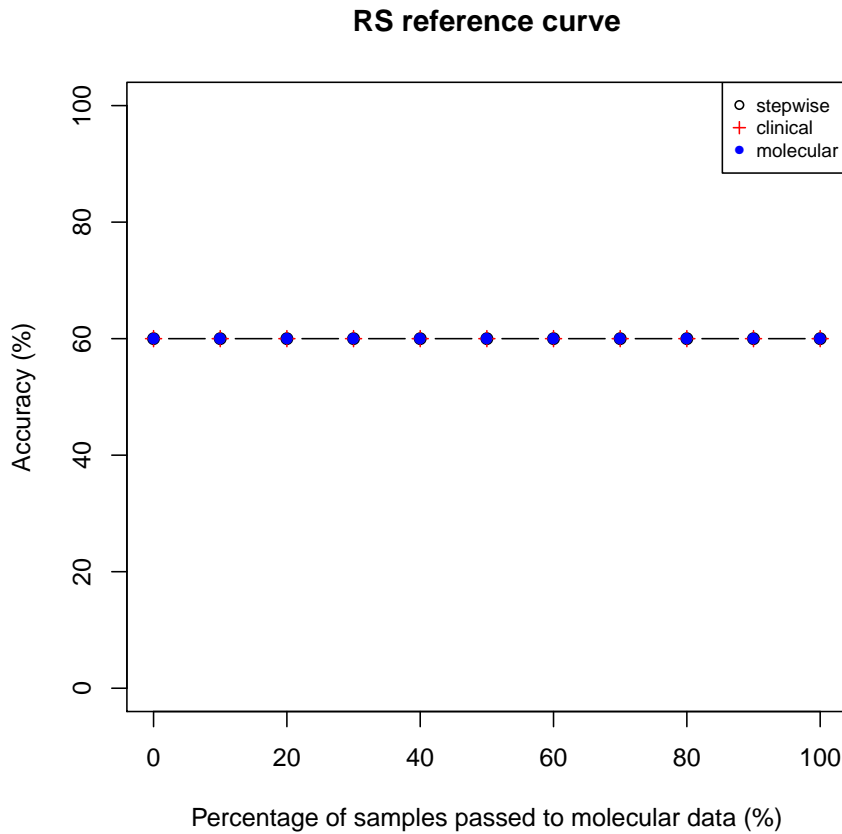
```
> RS <- RS.generator(pred.cli$P.train, pred.exp$P.train,
+ train.label, prox.cli$prox.test, prox.exp$prox.train,
+ "both")
```

```
> #observe the differences by ranking the RS values
> order(RS[, 1]) # from the ranking approach
> order(RS[, 2]) # from the proximity approach
```

6 Reference curve generation

To choose a RS threshold which produces reasonably high accuracy and at the same time keep large proportion samples at the first stage, we need a reference curve which shows how accuracy changes when different percentage of samples are classified with second stage data type. For this, we need pre-calculated RS of the test and predicted labels of this set from two data types independently.

```
> result <- Curve.generator(RS[, 1], pred.cli$P.test, pred.exp$P.test, test.label)
```



Accuracy plot can be generated by setting `plot.it = "TRUE"`. In this plot *X* axis denotes the percentage of samples classified with expression data and *Y* axis denotes the corresponding accuracy at that point. Note that since the tree construction inside the RF has randomness, so the resulting proximity matrix may not be the same in each run. This leads to different RS results in each time (but the difference will not be too large).

7 Assigning test samples to one of the two data types

Once we obtain the reference curve utilising the available samples for which both data sets are available, now we choose a RS threshold which pass specified percentage of samples to the second stage data type. We may use this threshold to all the incoming samples to decide whether we should measure its second stage data profile or simply classify it using the first stage data type.

```
> A <- Step.pred(RS[, 1], 30)
```

Here we allow 30% of samples can be classified with expression data. This function informs us the RS threshold which corresponding to the 30% re-classification and a vector of binary values indicate which samples are recommended to be classified with expression data (1) and vice versa (0).

References

- Obulkasim,A., Meijer, A.G. and van de Wiel, A.M. (2011). Stepwise Classification of Cancer Samples using Clinical and Molecular Data. *BMC Bioinformatics*, **12**, 422-424.
- Breiman, L., (2001). Random Forests. *Machine Learning*, **45**, 5–32.
- Pomeroy, S. L., Tamayo, P. and Gaasenbeek, M. (2002). Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression. *Nature*, **415**, 436–442.
- Boulesteix,A-L., Porzelius,C. and Daumer,M. (2008). Microarray-based Classification and Clinical Predictors: on Combined Classifiers and Additional Predictive Value. *Bioinformatics*, **24**, 1698-1706.